Linked List Implementation I

7 min

With the `Node` in hand, we can start building the actual linked list. Depending on the end-use of the linked list, a variety of methods can be defined.

For our use, we want to be able to:

- get the head node of the list (it's like peeking at the first item in line)
- add a new node to the beginning of the list
- print out the list values in order
- remove a node that has a particular value

Let's get started!

Note: Because the workspace is set up with spaces instead of tabs, you will need to use spaces to prevent Python from throwing an error. You can learn more about this here.

Instructions

1.

Within **script.py** in the pane to the right, create an empty `LinkedList` class.

Define an `.__init__()` method for the `LinkedList`. We want to be able to instantiate a `LinkedList` with a head node, so `.__init__()` should take `value` as an argument. Make sure `value` defaults to `None` if no value is provided.

Inside the `.__init__()` method, set `self.head_node` equal to a new `Node` with `value` as its value.

Hint

Don't forget to add `self` as a parameter for `.__init__()`.

Remember, to create a Python class with an `.__init__()` method:

```
class SomeClass:
  def __init__(self, some_attribute):
    self.some_attribute = some_attribute
```

2.

Define a `.get_head_node()` method that helps us peek at the first node in the list.

Inside the method, return the head node of the linked list.

Hint
Remember, `self.head_node` was defined in `__init__()`, so you can use it here!

A getter method should look something like:

```
def get_stuff(self):
  return self.stuff
```

script.py

```
# We'll be using our Node class
class Node:
  def __init__(self, value, next_node=None):
    self.value = value
    self.next_node = next_node

  def get_value(self):
    return self.value

  def get_next_node(self):
    return self.next_node

  def set_next_node(self, next_node):
    self.next_node = next_node

# Create your LinkedList class below:
class LinkedList:
  def __init__(self, value=None):
    self.head_node = Node(value)

  def get_head_node(self):
    return self.head_node
```